Scribe: A large-scale and decentralized application-level multicast infrastructure

Presented by Byungjin Jun
Overview

• Scribe: Application-level multicast infrastructure
  – Support large number of groups/members per group
  – Built upon Pastry

• Pastry: P2P location and routing substrate
  – Robustness
  – Self-organization
  – Locality
  – Reliability
Pastry (routing table in nodes)

- 128-bit nodeId
- The number of entries in a table
  - Routing table: \((2^b - 1) \times \lceil \log_2 N \rceil\)
    - \(N\): entire node in Pastry
    - \(b\): configuration parameter (typical 4)
  - Leaf set: \(l\)
    - \(l\): even integer parameter (typical 16)
    - \(l/2\) closest larger nodeIds
    - \(l/2\) closest smaller nodeIds

Fig. 1. Routing table of a Pastry node with nodeId 65a1x, \(b = 4\). Digits are in base 16, \(x\) represents an arbitrary suffix. The IP address associated with each entry is not shown.
Pastry (routing)

- Proximity-aware structured
  - Routes message to the Pastry node with closest key
- Average steps of Pastry routing
  - $\log_2 N$
- Delivery is guaranteed
  - Unless L/2 adjacent nodeIds fail
- Node addition / failure
  - Routing tables can be restored by exchanging $\log_2 N$ messages
  - To catch failure, periodically exchange keep-alive messages

Fig. 2. Routing a message from node 65a1fc with key d46a1c. The dots depict live nodes in Pastry’s circular namespace.
Pastry (Locality)

• Locality
  – Proximity metric
    • Reflecting “distance” between any pair of nodes
  – ”Short routes”
    • Route to the nearest node with a longer prefix match
  – ”Route convergence”
    • the average distance traveled by each of the two messages before their routes converge is approximately equal to the distance between their respective source nodes
Pastry (API)

• Pastry API
  – *nodeId = pastryInit(Credentials)*: local node joining or starting new pastry
  – *route(msg, key)*: route msg to the node with nodeId closest to key
  – *send(msg, IP-addr)*: send msg to the node at IP-addr
  – *deliver(msg, key)*
    • when a message is received and the local node’s nodeId is numerically closest to key among all live nodes
    • when a message is received that was transmitted via *send*, using the IP address of the local node
  – *forward(msg, key, nextId)*
    • just before a message is forwarded to the node with nodeId = nextId.
    • The application may change the contents of the message or the value of nextId.
    • Setting the nextId to NULL will terminate the message at the local node.
  – *newLeafs(leafSet)*: leaf set change
Scribe

- Scalable, decentralized application-level multicast
  - Large number of groups with wide range of group sizes
  - build a per-group multicast tree used for multicast
  - High rate of membership turnover
  - Best effort delivery but no particular delivery order
  - Any Scribe node can create, send messages to, and join many groups
  - Call API from Pastry, i.e. forward, deliver
Scribe API

• **create(credentials, groupId)**
  – creates a group with groupId. The credentials are used for access control.

• **join(credentials, groupId, messageHandler)**
  – causes the local node to join the group with groupId. All subsequently received multicast messages for that group are passed to the specified message handler.

• **leave(credentials, groupId)**
  – causes the local node to leave the group with groupId.

• **multicast(credentials, groupId, message)**
  – causes the message to be multicast within the group with groupId.
Implementation (Multicast message dissemination)

• Multicast tree
  – *rendezvous point*: root node of the tree
  – forwarder: Scribe nodes that are part of a multicast tree
    • *May or may not* be a members of the group
    • Maintains a children table containing an entry

• Multicast message dissemination
  – Locate *rendezvous point* for the group. [e.g., \texttt{route(multicast, groupId)}], and ask it to return its IP-addr.
  – The source caches the IP-addr and uses it for multicasts.
  – If the *rendezvous point* changes/fails, find the new one.
  – All multicast messages are sent from *rendezvous point*.
    • Rendezvous point perform Access Control
Implementation (Group Management)

• Create a group
  1. A Scribe node asks Pastry to route a create message using the groupld as the key. [e.g., route(create, groupld)]
  2. Pastry delivers the create message to a node that has its nodeId numerically closest to the groupld.
  3. Scribe’s deliver method is invoked and adds the new groupld to a list of groups. In addition, it also checks the credentials to ensure the group can be created.
  4. This node becomes the rendezvous point for the newly created group.

• Groupld
  – Hash of the group’s textual name concatenated with its creator’s name
  – Creator = rendezvous point?
Implementation (Membership Management)

• Joining a Group
  – Asks Pastry to route a join message with the groupld as the key. [e.g., route(join, groupld)]. The message is routed towards the rendezvous point.
  – Pastry invokes Scribe’s Forward method from every node on the route.
    • If the node is a forwarder for the group, it is added as a child.
    • If the node is not a current forwarder for the group, then it creates an entry for the group, adds the node as a child. Then the node becomes a forwarder by sending a join message to the next node along the root toward rendezvous point.
Membership Management (cont.)

• Leaving a Group
  1. Records a leaving of the node locally
  2. If there are no other entries in the children table, a leaving node sends a leave message to its parent
  3. 2. happens recursively until a node having children node after removing the departing child
Reliability (Repairing the multicast tree)

• Heartbeat message and failure tolerance
  – Non-leaf node sends it periodically to its children
  – If a child fails to receive it, considers parent is faulty and calls Pastry to route *join* message to the group.
  – Pastry route the message to a new parent, and tree fixed
  – Children table entries are discarded unless they send periodical message stating its desire to remain in the group

• Failure of *rendez-vous point*
  – info about group creator/access control list *rendez-vous point* has is replicated across the k closest children nodes
  – In case root fails, children detect it and join again, and a node having closest nodeId to groupId become the new root
  – Pastry route new *join* messages to the new root
Reliability (Providing additional guarantees)

• A simple mechanism for stronger reliability guarantee
  – forwardHandler(msg)
    • Before the node forwards a multicast message to its children.
    • Define that root assigns sequence number to each message such that recently multicast messages are buffered by root and each node.
  – faultHandler(msg)
    • When a node suspects its parent is faulty.
    • Adds last sequence number, n, delivered by the node to join message
  – joinHandler(msg)
    • After a new child is added to one of the node’s children tables.
    • Retransmits buffered messages with sequence numbers above n to the child node
Experimental Evaluation

- 5050 Routers, 100000 end nodes, 1500 groups
- Compare the performance of Scribe and IP Multicast
- Delay penalty
  - RMD
    - 50% of group ≤ 1.69
    - Max = 4.26
  - RAD
    - 50% of group ≤ 1.68
    - Max = 2
Experimental Evaluation (cont.)

- **Node Stress**
  - **End nodes** are responsible for maintaining membership information and forwarding and duplicating packets
  - Non-empty children tables per node: Avg=2.4, max=40
  - Children table entries per node: Avg=6.2, max=1059
  - Thin long tail result
Experimental Evaluation (cont.)

• Link stress
  – Environments
    • 1,035,295 links
    • 2,489,824 messages for Subscribe / 758,853 messages for IP Multicast
  – Mean number of messages per link
    • 2.4 for Scribe
    • 0.7 for IP Multicast
  – Maximum link stress
    • 4031 for Scribe
    • 950 for IP Multicast
Experimental Evaluation (cont.)

• Bottleneck remover
  – Bottleneck
    • Some nodes have less computational power or bandwidth available
    • Some nodes in the end of long tail of the number of children graph
  – Bottleneck remover
    • When overload is detected, selects the group consuming the most resources, and choose the child in this group that is farthest away
    • Parent drops the child by sending it a the children table
      – The child node measures delay between itself and other children
      – It compute the delay to get the parent node through each nodes
      – It send join message through the node providing the smallest delay
    – When a node receives a path containing itself, it send another join message to generate a new random route.
  – Drawback: increase link stress for joining
Experimental Evaluation (cont.)

- Scalability with many small groups
  - Average number of children entries per node
    - Scribe: 21.2 / naïve multicast: 6.6
  - Average link stress
    - Scribe: 6.1 / naïve multicast: 2.9 / IP multicast: 1.6
  - Scribe Collapse
    - Scribe multicast is not as efficient for small group
    - *Collapses* long paths in the tree removing nodes that are not members of a group and have only one entry in the group’s children table
    - Average number of children entries per node: 21.2 -> 8.5
    - Average link stress: 6.1 -> 3.3
### Related work

- A survey of application level multicast technique (C.K. Yeo, B.S. Lee, M.H. Er, 2004)

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Path length</th>
<th>Maximum outdegree</th>
<th>Average control overhead per node</th>
<th>No. of state entries per node</th>
<th>Failure tolerance</th>
<th>Group size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALM-CAN</td>
<td>Max: $O(dN^{1/d})$, Avg: $d/4N^{1/da}$</td>
<td>Constant of 2d</td>
<td>Constant of 2d</td>
<td>Constant of 2d</td>
<td>Localized failure</td>
<td>Large</td>
</tr>
<tr>
<td>Bayeux</td>
<td>Max: $O(\log_b N)$</td>
<td>$O(\log_b N)$</td>
<td>$O(b \log_b N)$</td>
<td></td>
<td>Single point of failure at root</td>
<td>Large</td>
</tr>
<tr>
<td>Narada</td>
<td>Max: unbounded</td>
<td>User defined to reflect bandwidth of user’s outgoing link</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td>Localized failure</td>
<td>Medium</td>
</tr>
<tr>
<td>Overcast</td>
<td>Max: unbounded</td>
<td>Unconstrained</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td>Single point of failure</td>
<td>Large</td>
</tr>
<tr>
<td>SCRIE</td>
<td>Avg: $O(\log^{b}_{2} N)$</td>
<td>$O(\log^{b}_{2} N)$</td>
<td>$O((2^{b} - 1)\log^{b}_{2} N)$</td>
<td></td>
<td>Localized failure</td>
<td>Large</td>
</tr>
</tbody>
</table>

*N* is the number of nodes in the overlay.

$a$ is the dimension of the Cartesian coordinate space of a CAN overlay.

$b$ is the base used in the Bayeux node ID.

Theoretically, worse case is created when $N - 1$ vertices form a circle and the $n$th vertex is in the center of the circle. However, the maximum outdegree is very small [12].
Conclusion

• Scribe is a fully decentralized and large-scale application-level multicast infrastructure built on top of Pastry.
• Scribe is designed to scale to large number of groups, large group size, and supports multiple multicasting sources per group.
• Scribe leverages the scalability, locality, fault-resilience and self-organization properties of Pastry and uses its advantages well.
• Experimental results show that Scribe can efficiently support large number of nodes, groups, and a wide range of group sizes compared to IP multicasting.